UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

Mobil alkalmazásfejlesztés - UI alapok - 2

Dr. Bilicki Vilmos

Szoftverfejlesztés Tanszék

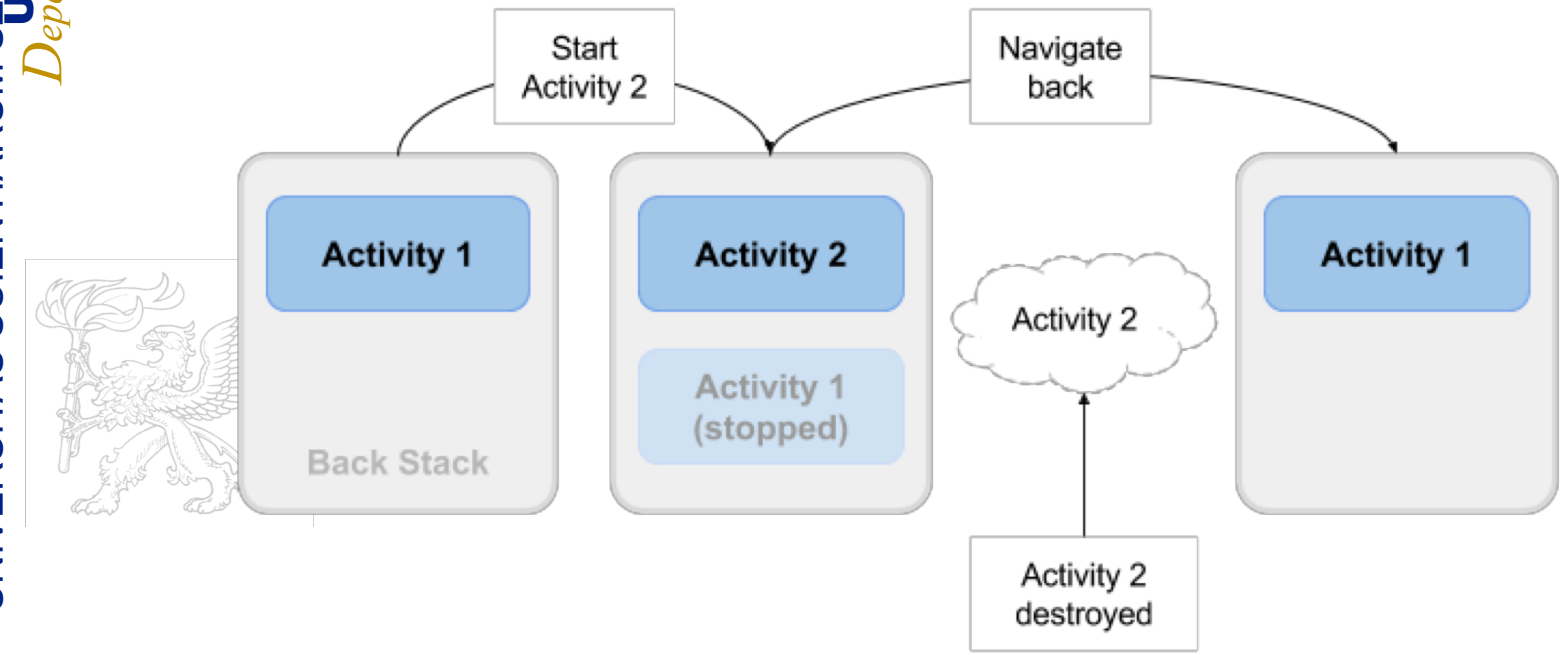
A fóliákhoz felhasznált anyagok:
Google: Android Developer Fundamentals (Version 2), https://developer.android.com/courses/fundamentals-training/overview-v2

Összefoglaló

- Aktivitás életciklus és állapotok
- Implicit Intent-ek
- Felhasználói interakciók
  - Gombok
  - Kattintható képek
  - Lebegő akciógombok
  - Általános mozdulatsorok
- Beviteli vezérlők

Acitivity életciklusa

- Egy irányított gráf az Aktivitás minden állapotával és a megfelelő callback-ekkel



Láthatóság és Activity állapotok

- Created (nem látható még)
- Started (látható)
- Resume (látható)
- Paused(részben látható)
- Stopped (rejtett)
- Destroyed (el lett távolítva a memóriából)

- Állapot változás lehet a felhasználói események vagy konfiguráció változás hatására (meg lett fordítva a a telefon)

Callback függvények

`onCreate(Bundle savedInstanceState)`—statikus inicailzáció

`onStart()`—amikor az Activity (képerny) láthatóvá válik

`onRestart()`—amikor az Activity leállt (meghívja az onStart()-ot)

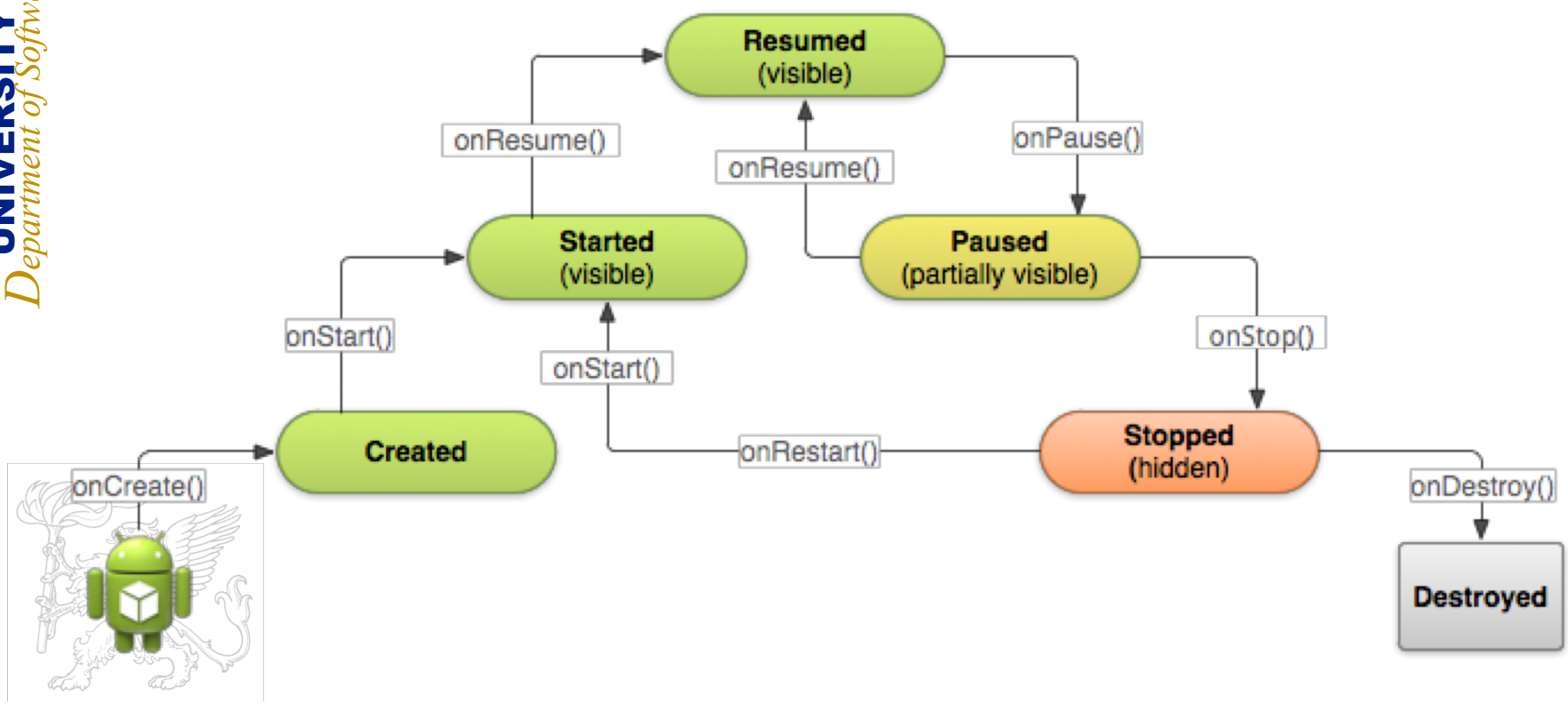
`onResume()`—fogadja a felhasználó interkacióját

`onPause()`—elindítja ELŐZŐ Activity-t

`onStop()`—nem látható, de minden állapota megvan

`onDestroy()`—az Activity megsemmisítése előtti utolsó hívás

Állaptok és callback-ek



Callback-ek megvalósítása

- Csak az onCreate()-t kell felülírni
  - onCreate() – amikor az Activity-t először létrehozzák, minden statikus beállítás elvégez (nézetek létrehozása), csak egyszer hajtódik végre, az onStart() után hajtódik végre
- A többit csak igény esetén
  - onStart – amikor láthatóvá válik, többször is meghívható, onResume, onStop után
  - onRestart - tranziens állapot (amikor leáll és utánna elindul)
  - onResume – elindul az Aktivitás (a verem tetjére kerül), a felhasználói bevitelt fogadja (onPause után)
  - onPause – az Activity részben látható, gyors implementáció kell ide (utánna onResume vagy onStop jön)
  - onStop – az Activity már nem látható (olyat tegyünk ide ami túl sok volt az onPause-be) (onRestart vagy onDestroy követi)
  - onDestroy – konfig változás, vagy user navigáció miatt (isFinishing), nem biztos, hogy ezt meghívja

Konfiguráció változás

- Érvénytelenné teszik a mostani elrendezést, vagy más az Activity-hez tartozó erőforrásokat
  - Eszköz rotálása
  - Nyelv változtatás
  - Több ablakos üzemmódra váltás
- Ekkor az Android
  1. Leálltíja az Activity-t (onPause,onStop,onDestroy)
  2. Újraindítja az Activity-t (onCreate,onStart,onResume)
- Activiy példány állapot
  - A futás közbeni állapotinformáció (animáció, szöveg,...)
  - Ez elveszik amennyiben rotáljál, nyelvet változtatnak, vissza gombot nyomnak, vagy a rendszer takarít

Aktivitás állapot mentés, visszaállítás

- Amit a rendszer elment
  - A nézetek állapotát (android:id) pl.: EditText-be bevitt szöveg
  - Az Activity-t elindító Intent és a bennne lévő extrákat
- Minden más mentéséről nekünk kell gondoskodnunk
  - Valósítsuk meg az Activity-n belül az onSaveInstanceState() metódust
    - Az Android környezet akkor hívja meg ha van esély arra, hogy az Activity-t megsemmisíti
    - Csak az adott példányhoz tartozó adatot menti el

onSaveInstanceState(Bundle outState)

```
@Override
public void onSaveInstanceState(Bundle outState) {
    super.onSaveInstanceState(outState);

    // Add information for saving HelloToast counter
    // to the to the outState bundle
    outState.putString("count",
                       String.valueOf(mShowCount.getText()));
}
```

Példány állapot visszaállítása

- Két módon kaphatjuk vissza a Bundle-t
  - Az onCreate(Bundle mySavedState) – fontos, hogy az UI minden adata minél gyorsabban vissza legyen állítva
  - Az onRestoreInstanceState(Bundle mySavedState) callback megvalósításával

onCreate()

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    mShowCount = findViewById(R.id.show_count);

    if (savedInstanceState != null) {
        String count = savedInstanceState.getString("count");
        if (mShowCount != null)
            mShowCount.setText(count);
    }
}
```

onRestoreInstanceState(Bundle state)

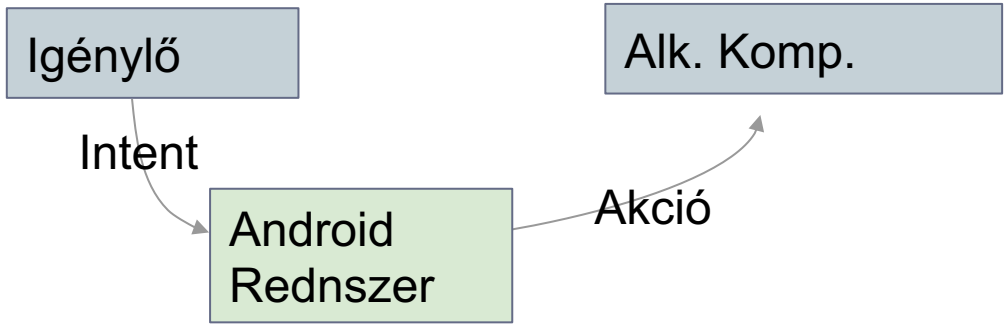
```
@Override
public void onRestoreInstanceState (Bundle mySavedState) {
   super.onRestoreInstanceState(mySavedState);

   if (mySavedState != null) {
       String count = mySavedState.getString("count");
       if (count != null)
           mShowCount.setText(count);
   }
}
```

Alkalmazás újraindítás

- Amikor az alkalmazást leállítjuk vagy újraindítjuk egy új alkalmazás viszonyban akkor az Activity példány minden adata elveszik
- Tartós mentés az alkalmazás viszonyok között: megosztott preferenciák, adatbázis

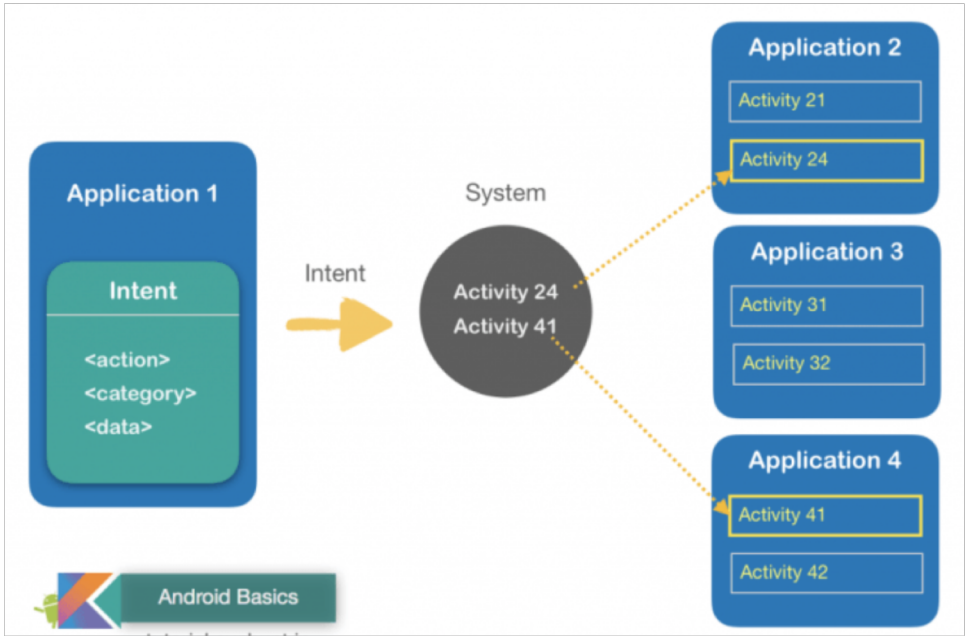
Implicit Intent-ek

- Intent
  - Egy művelet leírása
  - Üzenet objektum amellyel adott műveletet kér egyik komponens a másoktól



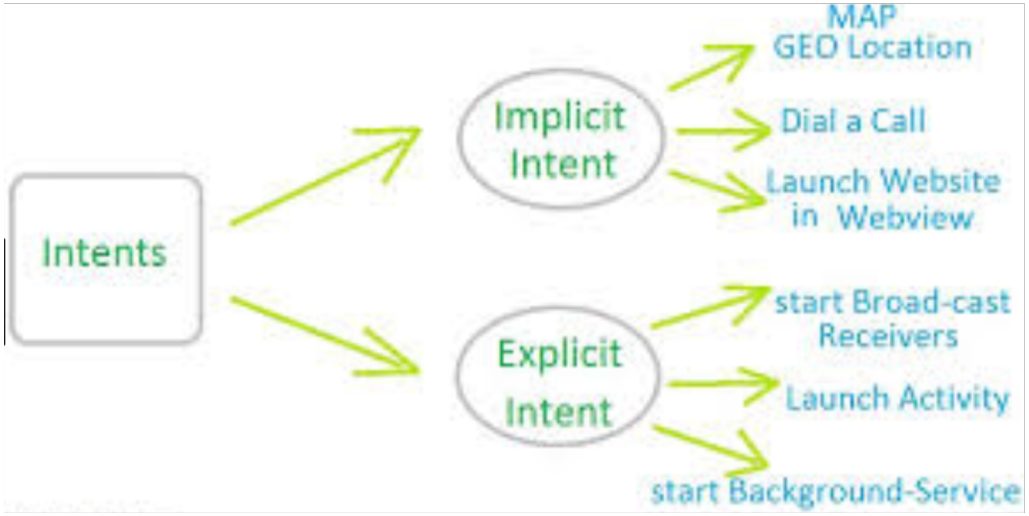
Intentek lehetőségei

- Activity elindítása
- Szolgáltatás elindítása
- Üzenetszórás kézbesítése



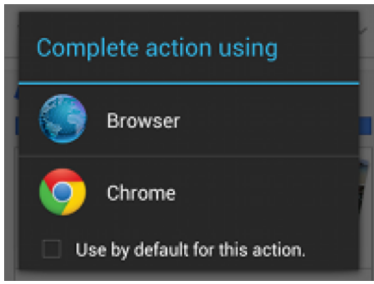
Explicit vs. Implicit

- Explicit – adott osztályhoz tartozó Activity-t indít
- Impicit – megkéri a rendszert, hogy keressen regisztrált kezelővel bíró Activity osztályt



Implicit Intent

- Képességei
  - Egy másik alkalmazás Activity-jének elindítása (pl.: cikk megosztása, ...)
  - Specifikál egy műveletet és a hozzá tartozó adatot
  - Nem adja meg a cél Activity osztályt csak a műveletet
- A rendszer feladata
  - A kérésnek megfelelő kezelőt keres
  - Amennyiben van több akkor az App Chooser-t teszi előtérbe



Részletek

1. Az Android Futtató környezet egy listában kezeli a regisztrált alklamazásokat
2. Egy alkalmazás az AndroidManifest.xml-ben adhatja meg szolgáltatásait
3. A futtató környezet megkapja a kérést és keres egy kiszolgálót (Intent filter)
4. Ha több is van akkor választani kell (App Chooser)
5. Elindítja a kiválasztott Activity-t

Implicit Intent küldése

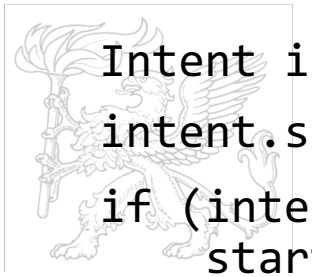
1. Intent létrehozása

```
Intent intent = new Intent(Intent.ACTION_CALL_BUTTON);
```

2. Activity elindítása

```
if (intent.resolveActivity(getPackageManager()) != null) {
        startActivity(intent);
    }
```

Adattal:



```
Intent intent = new Intent(Intent.ACTION_DIAL);
intent.setData(Uri.parse("tel:8005551234"));
if (intent.resolveActivity(getPackageManager()) != null) {
    startActivity(intent);
}
```

Adat mint URI

- Uri.parse("tel:8005551234")
- Uri.parse("geo:0,0?q=brooklyn%20bridge%2C%20brooklyn%2C%20ny")
- Uri.parse("http://www.android.com");

Példák

- Weboldal megnyitása

```
Uri uri = Uri.parse("http://www.google.com");
Intent it = new Intent(Intent.ACTION_VIEW,uri);
startActivity(it);
```

- Telefonszám felhívása

```
Uri uri = Uri.parse("tel:8005551234");
Intent it = new Intent(Intent.ACTION_DIAL, uri);
startActivity(it);
```

- Extrákkal

```
Intent intent = new Intent(Intent.ACTION_WEB_SEARCH);
String query = edittext.getText().toString();
intent.putExtra(SearchManager.QUERY, query));
if (intent.resolveActivity(getPackageManager()) != null) {
        startActivity(intent);
    }
```

Kategóriák

- CATEGORY_OPENABLE
  - Csak az adorr URI nyitható meg
- CATEGORY_BROWSABLE
  - Csak egy Activity fogadhatja meg amely támogatja a weboldal megnyitását

Implicit Intent típussal

```
Intent intent = new Intent(Intent.ACTION_CREATE_DOCUMENT);

intent.setType("application/pdf"); // set MIME type

intent.addCategory(Intent.CATEGORY_OPENABLE);

if (intent.resolveActivity(getPackageManager()) != null) {
startActivityForResult(myIntent,ACTIVITY_REQUEST_CREATE_FILE);
    }

...

onActivityResult()
```

Akció típusok és kezelőik

- Akció típus
  - ACTION_SET_ALARM
  - ACTION_IMAGE_CAPTURE
  - ACTION_CREATE_DOCUMENT
  - ACTION_SENDTO
  - …
- Ezeket kezelő alkalmazások
  - Alarm Clock, Calendar, Camera, Contacts
  - Email, File Storage, Maps, Music/Video
  - Notes, Phone, Search, Settings
  - Text Messaging and Web Browsing

Implicit Intent fogadása

- AndroidManifest.xml

```
<activity android:name="ShareActivity">
  <intent-filter>
    <action android:name="android.intent.action.SEND"/>
    <category android:name="android.intent.category.DEFAULT"/>
    <data android:mimeType="text/plain"/>
  </intent-filter>
</activity>
```

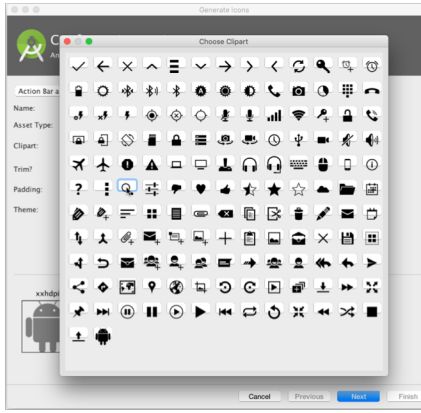
Intent szűrők

- action — Egy vagy több akció konstans
  - android.intent.action.VIEW — ACTION_VIEW intentek
  - android.intent.action.SEND — ACTION_SEND intentek
- category — tovább infó (kategóriák listája)
  - android.intent.category.BROWSABLE— web böngészővel nyitható
  - android.intent.category.LAUNCHER—Az Activity-t mind indító ikont mutatja
- data — Szűrés adat URI, MIME típus, ... alapján
  - android:scheme="https"—https enged csak
  - android:host="developer.android.com"—csak adott hostokról fogad intent-et
  - android:mimeType="text/plain"—korlátozza az elfogadható dokumentumokat

Gombok

- Nézet amely reagál a nyomásra, kattintásra, érintésre
- Szöveg, kép formájában jelzi, hogy mi fog történni
- Állapota: normál, fükuszban, tiltott, lenyomott, on/off

Válasz a gombnyomásra



- A kódban: OnClickLsietener
- XML-benandroid:onClick attribútum

```
<Button
android:id="@+id/button
_send"
android:layout_width="w
rap_content"
android:layout_height="
wrap_content"
android:text="@string/b
utton_send"
android:onClick="sendMe
ssage" />
```

android:onClick

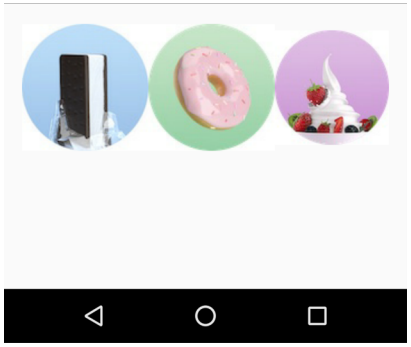
Esemény kezelő callback-kel

```
Button button = findViewById(R.id.button);

button.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        // Do something in response to button click
    }
});
```

Kattinható képek

```
<ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/donut_circle"
        android:onClick="orderDonut"/>
```

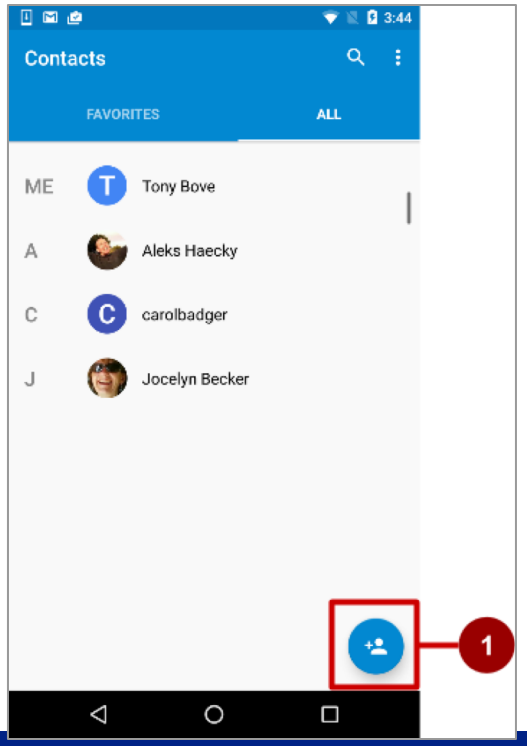
android:onClick



Floating Action Buttons (FAB)

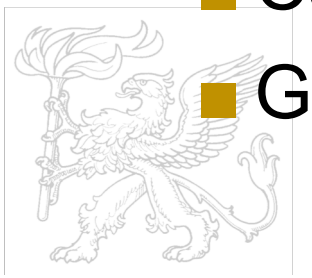
- Az alap aktivitás sablonnal

```
<android.support.design.widget.FloatingActionButton
        android:id="@+id/fab"
        android:layout_gravity="bottom|end"
        android:layout_margin="@dimen/fab_margin"
        android:src="@drawable/ic_fab_chat_button_white"
        .../>
```



Általános mozdulatsorok

- Érintés mozdulatosrok
  - Hosszú érintés
  - Dupla érintés
  - Húzás
  - Dobás
  - Csíppentés
  - Görgetés



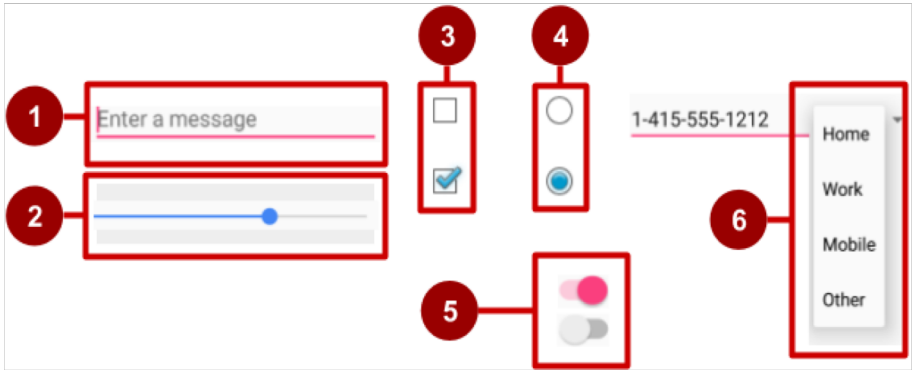
Az alkalmazás normál működéséhez ne használjuk ezeket!

Mozdulatsorok detektálása

- GestureDetectorCompat osztály a szokásos mozdulatsorokra
- MotionEvent osztály a mozgással kapcsolatos eseményekre
- Saját detektáló az alap esmeények alapján

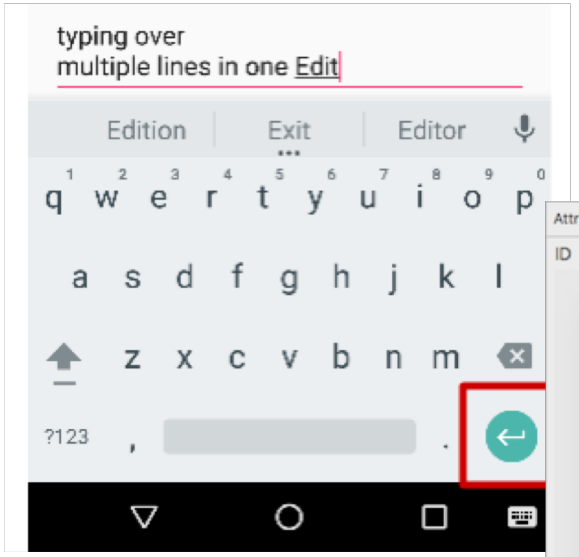
Felhasználói bevitel kezelése

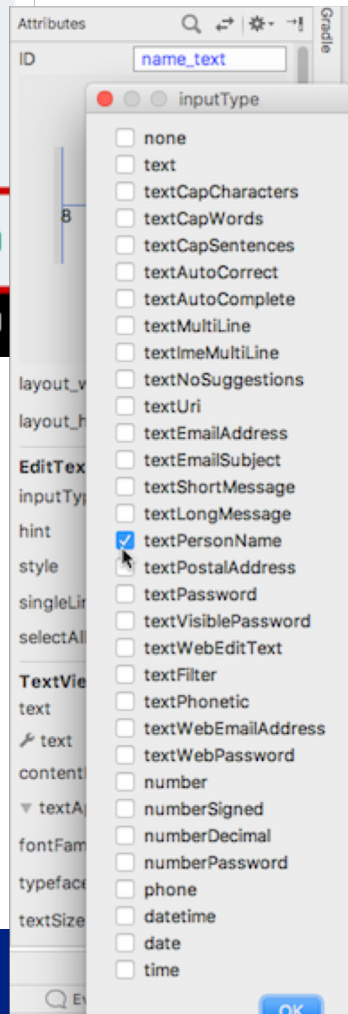
- Szabad szöveg, számok: EditText (1) (billentyűzettel)
- Választások: CheckBox (3), RadioButton(4), Spinner (6)
- Ki/Be kapcsolás: Toggle, Switch (5)
- Érték választása egy adott tartományból: SeekBar (2)



Szabadszöveges bevitel

```
<EditText
android:id="@+id/name_field"
android:inputType =
"textPersonName"
    ...
```





Hozzáférés az EditText objektumhoz

```
EditText simpleEditText =

findViewById(R.id.edit_simple);
```

- A tartalom lekérése

```
String strValue =

simpleEditText.getText().toString();
```

Fókusz

- View az alaposztály
- Az a View kapja meg a felhasználói eseményeket amelyik fókuszban van
- Csak egy View lehet fókuszban
- A fókuszt az alábbi módon jelöli ki az Android
  - A felhasználó megérinti az adott nézetet
  - Az alkalmazás vezeti a felhasználót az egyik beviteli mezőtől a másikig (Retrun, TAB, nyilak)
  - Meghívjuk a requestFocus() függvényt a kívánt nézetre
- Kattintható vs Fókuszálható
  - Kattinható: érintésre és kattintásra reagál
  - Fókuszálható: fókuszt kaphat

Összefoglaló

- Aktivitás életciklus és állapotok
- Implicit Intent-ek
- Felhasználói interakciók
  - Gombok
  - Kattintható képek
  - Lebegő akciógombok
  - Általános mozdulatsorok
- Beviteli vezérlők